Realtek iOS Simple Configure Wizard Guide

Version: 1

1. Introduction

This is the document describes how to use Realtek iOS Simple Config Wizard APP to configure WiFi Speaker and introduce simple config API.

2. Source Code Location and Description

2.1 Simple Config main application

SimpleConfig*: main package

2.2 Providing Resources

SimpleConfig\image: Bitmap files

2.3 Simple Config library

SimpleConfig\include\LibSimpleConfig: Simple config library

SimpleConfig\include\ZBarSDK: Library for QRCode scanner

3 Simple Config Library

3.1 Description

Simple Config for IOS contains two major interfaces: **PatternFactory** and **SimpleConfig**.

PatternFactory is the underlying class which implements Pattern 2 and Pattern 3. SimpleConfig is an API, which inherits from PatternFactory. SimpleConfig supplies developers with external APIs for Simple Config further development.

3.2 PatternFactory

Table 3-1 Details about PatternFactory are as follows

| Member Variables of Interface PatternFactory |
Name	Data type	Description
udpSocket	AsyncUdpSocket	Socket to send multicast packets, such as configure and discover packets
controlSocket	AsyncUdpSocket	Socket to send unicast packets, such as control packets.
have_pin	unsigned int	Check whether has user PIN or not, 0 for no and 1 for yes. If have_pin is set, will use m_pin for configure. Otherwise, will use default_pin for configure. Will be set in API *rtk_sc_set_pin* and *rtk_sc_set_default_pin* automatically.
m_pin	NSString *	Store user PIN. Should be set through API *rtk_sc_set_pin.*
m_ssid	NSString *	Store SSID that will be used for configuration. Should be set through API *rtk_sc_set_ssid.*
m_psw	NSString *	Store password for m_ssid. Should be set through API *rtk_sc_set_password.*
default_pin	NSString *	Store the default pin, which will be used when have_pin is 0. Should be set through API *rtk_sc_set_default_pin.*
send_times	int	Store the send times per round while configuring devices. Can be set directly.
ack_host	NSMutableArray *	Store the devices' MAC address, which are successfully configured.
dev_connected_list	NSMutableArray *	Store the devices' information, which are discovered.
m_control_data	NSData *	Store the UDP packets of device control data.
m_discover_data	NSData *	Store the UDP packets of device discover data

3.3 SimpleConfig (External API)

SimpleConfig supplies the API to developers.

Member Variables of Interface PatternFactory		
Name	**Data type**	**Description**
pattern	PatternFactory	Instantiation for interface PatternFactory.
Member Functions of Interface PatternFactory(A): Device Configure		
API		**Description**

-(id)initWithPattern: (unsigned int)index
            flag: (unsigned int)flag
            name: (NSString *)name
		Initial function, will initial pattern as well.
		Index: pattern number;
		flag: use encryption or not;
		name: pattern name.

- (void)rtk_sc_set_pin: (NSString *)pin		Set user PIN as pin, which will be used when configuring devices. Will set *pattern.m_pin*, *pattern.have_pin*.
- (void)rtk_sc_set_ssid: (NSString *)ssid		Set SSID as ssid, which will be used when configuring devices. Will set *pattern.m_ssid.*
- (void)rtk_sc_set_password: (NSString *)psw		Set SSID password as psw, which will be used when configuring devices. Will set *pattern.m_psw.*
- (void)rtk_sc_set_ip: (unsigned int)ip		Leave empty currently.
-(void)rtk_sc_gen_random		Generate random values to secure configure packets.
- (void)rtk_sc_build_profile		Firstly, build the plain data which is used for device configuring. Secondly, generate keys and check flag set in *initWithPattern*, if needs encryption, encrypt the plain data.
- (void)rtk_sc_send		Send device configure packets. Note: Will only send one round with *pattern.send_times* times.
- (int) rtk_get_connected_sta_num		Return numbers of devices which are successfully configured.
- (NSMutableArray *) rtk_get_connected_sta_mac		Return the devices' MAC address which are successfully configured.
- (void)rtk_sc_stop		Should be called after developer stops calling *rtk_sc_send*

- (void)rtk_sc_exit	Leave empty currently.
- (NSString *) rtk_sc_get_default_pin	Get current default PIN.
- (void) rtk_sc_set_default_pin:(NSString *)pin	Set default PIN as pin.

Member Functions of Interface PatternFactory(B): Device Control	
-(void) rtk_sc_clear_device_list	Clear all devices discovered in former operations.
- (NSData *)rtk_sc_gen_discover_packet	Generate device discover UDP packets, return in NSData* format.
- (void) rtk_sc_send_discover_packet: (NSData*)data ip:(unsigned int)ip;	Send device discover packets data to ip using UDP. (Multicast). Note: **exclude** rename device operation.
-(NSMutableArray *)rtk_sc_get_discovered_devices	Get devices array which are discovered.
- (void) rtk_sc_set_control_pin: (NSString *)pin	Set device PIN for device control operations.
- (NSData *)rtk_sc_gen_control_packet: (unsigned int)control_type;	Generate device control UDP packets, whose control type is control_type, see in Chapter 1.4 constant values.
- (void) rtk_sc_send_control_packet: (NSData *)data ip:(unsigned int)ip;	Send device control packets data to ip using UDP. (Unicast) Note: **exclude** rename device operation.
- (NSData *)rtk_sc_gen_rename_dev_packet: (NSString *)dev_name	Generate rename device UDP packets. New name is dev_name.
- (void) rtk_sc_send_rename_dev_packet: (NSData *)data ip:(unsigned int)ip	Send rename device packets data to ip using UDP (Unicast).
- (int) rtk_sc_get_control_result	Get control operations result. 1 for success, 0 for failed, other for timeout.
- (void) rtk_sc_reset_control_result	Should be called immediately after get control results.

3.4 Constant Values and misc. structure

1. dev_info: stores the discovered device information, used in device control.

```
typedef struct _dev
{
    NSString *mac;
    unsigned char type[2];
    unsigned int ip;
    NSString *dev_name;
}dev_info;
```

2. Control type: defines device control operation types

```
typedef enum{
    SC_DELETE_PROFILE=1,
    SC_RENAME_DEV=2,
}SC_CONTROL_TYPE;
```

4 Simple Config Working Flow

Simple Config can be used to:

　　1. Configure a client device;

　　2. Discover devices;

　　3. Control devices, including delete profile and rename device.

The working flow of each function will be introduced in this chapter, so that developer can call API correctly.

4.1 Device configuration

The working flow of device configure is shown as Fig 4-1.

```
┌─────────────────────────┐
│     rtk_sc_set_pin      │
└─────────────────────────┘
            │
            ▼
┌─────────────────────────┐
│     rtk_sc_set_ssid     │
└─────────────────────────┘
            │
            ▼
┌─────────────────────────┐
│   rtk_sc_set_password   │
└─────────────────────────┘
            │
            ▼
┌─────────────────────────┐
│    rtk_sc_gen_random    │
└─────────────────────────┘
            │
            ▼
┌─────────────────────────┐
│   rtk_sc_build_profile  │
└─────────────────────────┘
            │
       ┌────┴─────────────────────┐
       ▼                          │
   ╱stop configuring?╲   N   ┌─────────────┐
   ╲                ╱ ─────> │ rtk_sc_send │
       │                     └─────────────┘
       │ Y
       ▼
┌─────────────────────────┐
│      rtk_sc_stop        │
└─────────────────────────┘
            │
            ▼
┌──────────────────────────────┐
│  rtk_get_connected_sta_num   │
└──────────────────────────────┘
            │
            ▼
┌──────────────────────────────┐
│  rtk_get_connected_sta_mac   │
└──────────────────────────────┘
```
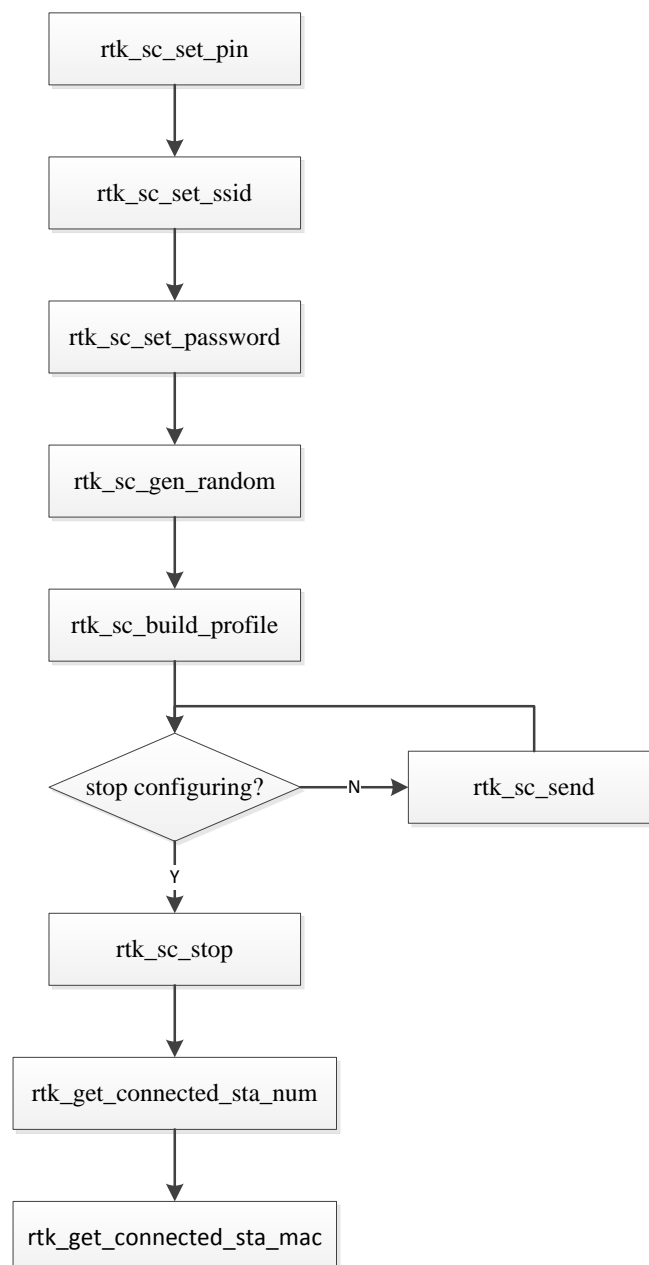
Note that it's developer's duty to decide when to send configure packets and when to stop sending. For example:

```
[self.simpleConfig rtk_sc_build_profile];
while(CONDITION){ [self.simpleConfig rtk_sc_send];}
[self.simpleCofnig rtk_sc_stop];
```

4.2 Device discovery

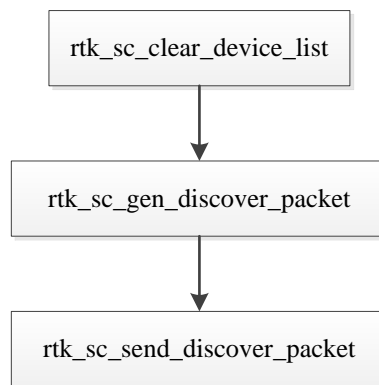The working flow of device configure is shown as Fig 4-2.

Figure 4-2 Device discovery working flow

Developers can call API *rtk_sc_get_discovered_devices* to get the discovered devices' information. Note that the returned NSMutableArray * stores structure dev_info, so developers should call such codes to get each object:

```
NSMutableArray *dev_list =[self.simpleConfig    rtk_sc_get_discovered_devices];
int idx = user_defined_idx;
dev_info *dev;
NSValue *dev_val = [dev_list objectAtIndex:idx];
[dev_val getValue:&dev];
```

4.3 Device control

Device control includes two parts: delete profile and rename device. They all needs user to input PIN first. Additionally, rename device requires user to input device new name. So rename device has a different API with the former two.

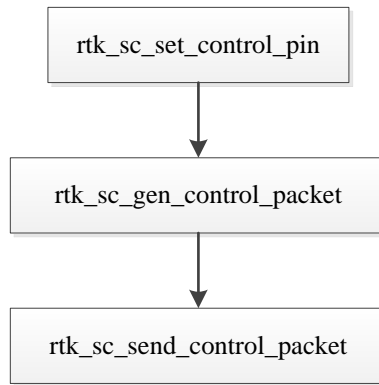This general working flow is shown as Fig 4-3.

```
                    ┌─────────────────────────────┐
                    │    rtk_sc_set_control_pin    │
                    └─────────────────────────────┘
                                   │
                                   ▼
                    ┌─────────────────────────────┐
                    │  rtk_sc_gen_control_packet   │
                    └─────────────────────────────┘
                                   │
                                   ▼
                    ┌─────────────────────────────┐
                    │  rtk_sc_send_control_packet  │
                    └─────────────────────────────┘
```

Figure 4-3 Delete profile working flow

Developers can use API *rtk_sc_get_control_result* to get the control result. It's important to note that once the API is called, developer must call *rtk_sc_reset_control_result* to reset the result, so that it's working good next time.
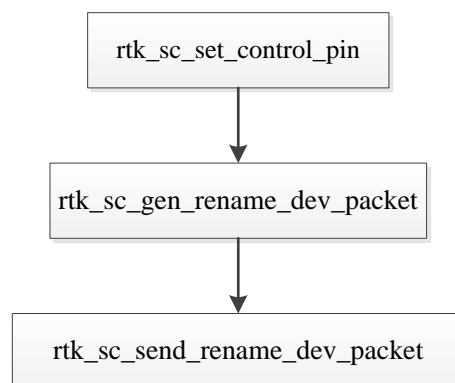

Rename device working flow is shown in Fig 4-4.

```
                    ┌─────────────────────────────┐
                    │    rtk_sc_set_control_pin    │
                    └─────────────────────────────┘
                                   │
                                   ▼
                  ┌───────────────────────────────┐
                  │  rtk_sc_gen_rename_dev_packet  │
                  └───────────────────────────────┘
                                   │
                                   ▼
                 ┌─────────────────────────────────┐
                 │  rtk_sc_send_rename_dev_packet   │
                 └─────────────────────────────────┘
```

Figure 4-4 Rename device working flow

Also developers can use API *rtk_sc_get_control_result* to get the control result.

5 Reference

N/A